Computer Science

Dialog Structure and Plan Recognition in Spontaneous Spoken Dialog

Sheryl R. Young

June 1993
CMU-CS-93-199

Carnegie Mellon

Dialog Structure and Plan Recognition in Spontaneous Spoken Dialog

Sheryl R. Young

June 1993

CMU-CS-93-199

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

This research was sponsored by the Department of the Navy, Naval Research Laboratory under Grant No. N00014-93-1-2005; and by the Department of the Navy, Office of Naval Research under Grant No. N00014-93-1-0806.

The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of NRL, ONR, or the U.S. Government.

AD NUMBER	DATE	DTIC ACCESSION NOTICE
1. REPORT IDENTIFYING INFORMATION		**REQUESTER:**
A. ORIGINATING AGENCY CARNEGIE MELLON UNIV		1. *Put your mailing address on reverse of form.*
B. REPORT TITLE AND/OR NUMBER CMU-CS-93-199		2. *Complete items 1 and 2.*
C. MONITOR REPORT NUMBER		3. *Attach form to reports mailed to DTIC.*
D. PREPARED UNDER CONTRACT NUMBER N00014-93-1-2005 / N00014-93-1-0806		4. *Use unclassified information only.*
2. DISTRIBUTION STATEMENT (A) UNLIMITED DISTRIBUTION		5. *Do not order document for 6 to 8 weeks.*
		DTIC:
		1. *Assign AD Number.*
		2. *Return to requester.*

DTIC Form 50 DEC 91 PREVIOUS EDITIONS ARE OBSOLETE

Abstract

In real spoken language applications, speakers interact spontaneously and frequently diverge from the task at hand by initiating various types of domain, application or environmentally related subdialogs. We claim that unconstrained, task-oriented spontaneous spoken dialog is structured and predictable in spite of such phenomena as spurious topic changes and subdialogs. The discourse structure for any specific dialog is derived from the structure of the task, contextual constraints derived from prior interaction and the characteristics of a finite set of discourse plans responsible for subdialogs and topic changes. This paper describes a preliminary model of discourse structure and plan recognition for spontaneous spoken discourse that has been implemented and evaluated on a 5000 utterance test corpora drawn from two distinct spoken language applications. The model dynamically constrains a speech recognizer, simplifies the process of inferring meaning from a spontaneous spoken utterance and accounts for the subdialog phenomena observed. We describe these discourse plans, constraints on their occurrence and content, and their representation and processing. The model processes all subdialog phenomena using a domain plan tree, a current focus stack and a set of domain tree traversal algorithms.

1. Introduction

Discourse has been studied from the point of view of *plan recognition* [6, 8, 3], speech acts [1] and domain independent properties of discourse structure [5, 2]Inferring a speaker's underlying plans and intentions assists in interpreting both what is stated and what is implied and intended. The advent of large corpora of real, naturally elicited dialogs from multiple application domains have provided many examples of spontaneous spoken discourse. They have enabled researchers to identify, characterize and model domain independent discourse properties of task-oriented dialogs. Spoken language system [SLS] applications enable researchers to empirically evaluate their discourse models and plan inference and tracking methods for thoroughness, coverage and explanatory utility. This is because SLS permit evaluation of both language understanding capabilities and any associated spontaneous speech recognition effects that result from specifying how prior discourse will constrain the next actions a speaker can take, or what can be said next.

Domain Plans. Until recently, SLS discourse models focused solely on domain plans. These systems coupled domain plans with *semantic and pragmatic knowledge and inferencing* procedures to compute set of "next actions" [10, 7, 9]. Domain plans refer to utterances where a plan step is described or implemented. For each application, it is possible to develop an application specific, generic set of hierarchically organized plans that include all plans that could be executed during an interaction. Development of a domain plan tree is normally guided by the nature of the problem solving or information seeking task and by the structure of the task itself, including the semantic relations among objects, attribute, actions, etc.

The use of domain plans to represent task structure, and plan inference and tracking was first introduced in the late 1970's [4] for interpreting task-oriented dialogs. This work illustrated that topics of conversation in a task-oriented dialog could be anticipated by representing the semantic attributes of the task and tracking the progress of the plans of the conversants as the task was executed. By associating specific objects, attributes and actions with each plan step a speaker could execute during a dialog, similar objects could be uniquely identified and objects in focus could be computed.

More recently, hierarchically structured domain plans have been shown to be extremely powerful tools for significantly enhancing overall system performance by improving a SLS's ability to infer utterance meaning in applications that involve information seeking, problem solving and task oriented dialogs. These systems adopt a plan-tracking and inference approach to natural language understanding and are usually able to circumscribe or predict the content of a "next utterance" or input by eliminating interpretations that would be meaningless, redundant or inconsistent from consideration. Generally, these predictions are derived by tracking currently active domain plans and applying semantic and pragmatic constraints computed by propagating contextually appropriate information obtained or implied by earlier interaction. When incorporated in a general interactive, integrated or feedback architecture, these predictions significantly enhance overall system performance further because they can be used to dynamically constrain the active lexicon used to process an incoming utterance. By dynamically modifying the lexicon / language model and grammar, *eliminating many words from consideration during* the speech recognition, the search space for words is significantly reduced and recognition performance is thereby significantly improved.

Discourse Plans. However, prediction capabilities based solely upon a representation of domain plans cannot cope with spontaneous discourse produced when speakers are unconstrained and speak naturally and spontaneously. The traditional plan tracking approach runs into trouble when faced with subdialog phenomena, inclusive of clarifications, corrections and topic changes. To deal with situations where a one cannot track the dialog exclusively by looking at the domain plans for accomplishing the task in a hierarchical, stepwise format alone, Litman introduced the notion of discourse plans [6]. Discourse plans specify the type of action a user can execute (verbally) in the dialog. They include such actions as "continue" the current domain plan, "begin a subdialog", for example, to "clarify" the last input or database / other speaker response or "initiate a topic switch" to ask a question about the currently active external environment (e.g. "*Where can I find a newsstand?*" or "*Display that again*"). Discourse plans, or the types of actions a user can initiate to control the dialog, are domain independent. When speakers interact naturally, even cooperative users tend to digress and clarify. Discourse plans describe these digressions as well as the normal domain plans or domain relevant discussions in terms of how these actions effect or control the dialog. At each point in the discourse interaction, only a limited set of discourse plans can be executed. To process spontaneous spoken dialog, it is necessary to take discourse plans as well as domain plans into account.

Overview. We claim that spontaneous spoken dialog has a predictable structure that is defined by the properties of discourse plans and their interaction with the domain plans for an application. These structural properties dictate "what can occur when". They indicate what topics can be "switched to" at any given point in the dialog, what information can be clarified, and what types of subdialogs can be initiated at a given point in time. Further, they place constraints on the objects and attributes available for reference. We have implemented our model of discourse structure in a SLS [12] that exploits these properties to dynamically constrains both the speech recognition and utterance comprehension processes.

In this paper, we describe our model of spontaneous spoken discourse for spoken language systems. We identify and describe the range of discourse plans observed in the training sets from four distinct SLS applications and the conditions under which they can appear. Specifically discussed are continuation subdialogs, clarification, confirmation and correction subdialogs, and topic changes and resumptions. Our descriptions include what's available for reference and implications for future utterances. The paper presents our unified algorithm for recognizing discourse and domain plans and using them for predicting future actions and propagating constraints derived from information introduced earlier in the dialog.

Basically, our system "predicts" the types of "next actions" or discourse plans that can be executed next and constraints on the sets of parameters associated with each of the potential actions. We try to answer the question *"How does prior discourse constrain what can happen next?"* The system has been evaluated by processing over 5000 spontaneous spoken utterances collected from two domains. The first domain is composed of mixed initiative dialogs where subjects order lunch from a pizzeria. We have 2,600 utterances in this test set. The second domain is from ARPA's ATIS or air travel information domain. These are user initiated dialogs where the computer responds by displaying information contained in the airline's OAG database. We have 2350 test utterances from ATIS.

2. Discourse Plans: The Taxonomy and Basic Processing

In real applications, speakers frequently diverge from the task at hand by initiating various types of domain, application or environmentally appropriate or related subdialogs. These behaviors range from initiating a clarification subdialog, to either modify their last input or to request an explanation of the response, to requesting information about their external environment, as in *"scroll the screen down"* or *"where can I buy a newspaper?"*. Based upon our corpora, we have developed a taxonomy of discourse plans.

The taxonomy is based upon the function served by the subdialog or discourse plan in the larger interaction. While there have been a few other attempts at categorizing discourse plans [6, 2], ours differ in that they are computationally implemented in an operational spoken language system and are based primarily upon the use of a domain plan tree and a current focus stack. Our taxonomy focuses on how the subdialog changes the on-going dialog. Our basic types are generated by grouping those phenomena that can be computationally represented and processed in a distinct manner. Here, we indicate the types of phenomena included in each category and provide illustrative examples. We describe their prevalence in our test domains, when they can occur, what information is available for reference and how each is represented. Finally we describe what information is propagated to the "main dialog" and available for future reference. We have identified the following discourse plans:

- Discourse Plans:
    - Continue Domain Plan (56 - 58% utterances)

- Begin Subdialog:
  - Clarification (2% - 10% utterances)
  - Confirmation (<1 - 2.5% utterances)
  - Correction
    - correct due to confirmation (1% utterances)
    - correct due to plan failure (24-35% utterances)
- Initiate Topic Change:
  - New domain plan (4.5 - 12% utterances)
  - External Context (<1% utterances)
  - Historical Context (0% utterances)

2.1. Clarifications

Clarificational subdialogs either clarify the user's question / statement or the response obtained. When the response is clarified, the user or machine can ask about the range of values acceptable, the meaning of some item contained in the response, an attribute of a value that is part of the range of acceptable responses, or an attribute of one of the items named in the response. Often, clarifications are used to obtain information required for performing the required task. For example, consider the following:

`Would you like any toppings?`
Brie, camembert, mushrooms and olives
`We don't have brie and camembert`
*******Do you have cheddar?*
`**Yes`
*******How much is pepperoni?*
`**All toppings are 75 cents`
OK, mushrooms, olives, cheddar and pepperoni
`**Black or green olives?`
*******Black*

This example illustrates that clarificational subdialogs can be nested and initiated by either conversational participant. Clarification dialogs were prominent in our data. Their content does not affect the domain plan tree except in those cases where a speaker clarifies their input by asking an essentially different question (or giving a different response) that opens a different node in the domain plan tree. Normally, the clarification serves to provide additional information. We represent the content of clarifications and nested clarifications in a focus stack as illustrated in Section 4. The only information available for reference in a clarificational subdialog are the objects and attributes contained in the immediately preceding turn. Hence, when a nested clarification is initiated, the only information available for reference is information in the last clarification turn. No information from a clarificational subdialog is propagated into the "main discourse" and the domain tree is not modified.

2.2. Confirmations

Confirmation subdialogs can only occur at the end of a subtask, or when a domain subtree is complete. For example, in the ordering scenario (see Figure 1), pizza specifications can be confirmed at three possible places in the dialog, 1) immediately after the pizza is specified, 2) when all the food is ordered, or 3) when the dialog is nearing completion and the entire order has been completed. A confirmation can be initiated by either conversational participant to verify that they have understood correctly. Each item in the applicable completed unit can be verified and is available for reference. Confirmation are often followed by correction subdialogs, and occur when the information to be confirmed is incorrect.

We process confirmation subdialogs using the domain tree alone. All completed nodes in the applicable subtree are temporarily re-activated by a clarification until they have been discussed or the confirmation is complete. The following example illustrates a confirmation subdialog responded to with a correction subdialog (starred).

```
OK flight 49 on US Air leaves Pittsburgh
at 5:07p.m. arrives Los Angeles at 8:05 p.m.
on November 15.  Cost is $1159.
```
No that was for $629.
```
**All seats are sold for that fare class.
```
Do you have any seats on an earlier flight for $629?
```
**There's one seat left on the 9:05 am flight
```

2.3. Corrections

Correction subdialogs are initiated under two conditions, when a confirmation fails or when there is a plan failure. These two are grouped together because both serve to re-activate a completed portion of the domain tree. Plan failures are easily detected, normally the user will encounter a null database response or be explicitly informed of a plan failure. For example, they will not be able to get a cheap fare on a dinner flight, or there will be insufficient resources in a resource limited problem solving domain. Similarly, following specification of toppings and size in a pizza ordering domain, the user finds out that you cannot get a small thick crust pizza, only medium and large sizes come with extra thick crusts. Plan failures occur when a user cannot fulfill all their requirements simultaneously. They are followed by a re-planning phase where the user must prioritize goals and then abandon one or more.

In our system, when a plan fails, all the specifications up to and including the point of the failure become re-activated in the domain tree. On the other hand, when a correction is initiated in response to a confirmation failure, the relevant nodes are already activated and only the node where the failure occurred and nodes that are causally related to it are available for reference and reexamination during the correction phase. So, in the above pizza example, the toppings, crust and size nodes would all become active until the user modified their specification. Correction subdialogs should not be confused with clarification subdialogs when a speaker clarifies and corrects the interpretation of their last input. Corrections only follow confirmations or plan failures.

2.3.1. Changing Topics

We have identified three types of topics switching phenomena:

1. ***Domain Goal*** - when a second or additional domain goal is initiated,
2. ***External Environment*** - when the user asks a question or makes a request about the immediate, (modelable) external environment, and
3. ***Historical Context*** - when the user switches topics to resume or follow up on a discussion that took place at an earlier point in time.

The system can process the first two. Consider the following:

Show flights from Pittsburgh to Boston
Show flights from Boston to Denver
List flights from Denver to Pittsburgh

```
I need a ticket for flight 286 to Boston
```
That will be $358 one way
<conversation continues>
```
**Where can I buy a pack of cigarettes?
```
Just past the banking machines on the left is a newsstand
```
OK, and which way to gate 21?
```
That's gate 36.
We will begin boarding in 15 minutes.
```
**Do I have time to go to the newsstand?
```

To process topic changes manifested as additional goals to be fulfilled (as in the first example), the system generates a second or additional instance of a domain tree and places the new domain topic on the active focus list. Hence, after processing the first example, there would be three active main plans on the focus stack and three instantiations of the domain plan tree. When one of the goals is completed, it is assumed that the speaker will return to the other ones. Generally, multiple topics are introduced in the beginning of a dialog and then one is pursued to completion before the other(s) are begun. (In fact, we have not seen a single instance where multiple topics have been introduced at any place other than the beginning of a dialog. Further, we have not seen a single instance where an introduced topic has not been pursued later in the dialog.)

To process topic changes where a speaker initiates a query about the immediately surrounding environment, it is necessary to directly model the environment. Today's technology does not permit us to model all attributes in a face to face environment, such a what a person is wearing or where they are gesturing. However, we can model the standard external environment of the system user and of a domain plan. For example, we can model the fact that a user is interacting with a terminal screen, or that a user is standing in at an airport ticket purchasing desk that is located in the main terminal building along with restaurants, newsstands, bars, etc. By activating the context in which an interaction takes place and the surrounding context of the query (e.g. questions about gates refer to the gate area, outside the main terminal) we can anticipate most external environment requests. Our data indicates that these requests occur immediately after a subtask is completed (e.g. once ticket purchase exchange is complete, ask about main terminal area), or when there is a change in the external environment (i.e. new information is printed on the screen and the user asks for a redisplay).

3. Data Structures and Processing

Our dialog system relies upon a domain specific structured knowledge base that contains a representation of the plans that can be executed in the application domain and a focus stack. The knowledge base must be generated for each application domain. However, the algorithms responsible for plan inference and tracking, constraint propagation, general inferencing and for processing subdialogs, plan failures and other discourse plans are constant across applications. The basic idea underlying the system is that by tracking all information communicated it is possible to infer speaker goals and plans and tracking progress. Further, by tracking progress, it is possible at each point in the dialog to specify or predict the types of discourse actions that can be taken, their relative probabilities (must be computed separately for each application) and constraints upon the content of each of the applicable discourse plans. These "predictions" can then be used for better inferring utterance meaning, for detecting misrecognitions and to dynamically generate grammars for reprocessing misrecognized input [12] or to guide the initial recognition process.

The domain knowledge base represents all objects, attributes, values, plans, goals and the environment in which the actions and plans are executed. It uses a standard frame-based representation to represent knowledge and is composed of four componenent knowledge bases each represented in a "plane". Information about objects, attributes and values are stored in "one plane" of the knowledge base, action and event information in a second, information about plans is stored in a third and goals are stored in a fourth plane. Within a plane, we have standard tangled inheritance networks and multiple relations among frames and frame slots. However, inheritance and inferencing across planes is somewhat more structured. For example, actions involve objects, their attributes and values. An action can activate a plan step. Plans contribute to the satisfaction of goals. In this way, we can limit spurious inferences and represent actions differently than plans which are different than objects and attributes, etc. Consider the simple task of ordering lunch from a pizzeria. The object plane of the knowledge base represents pizza and that pizzas come in different sizes, whose values are number of slices or diameter, have different types of crusts and have a set of toppings, including the defaults of tomato sauce with spices and mozzarella cheese. We also encode that the pizza is an edible object as are the toppings. The toppings include meats, vegetables, cheeses and fish. Since a pizza is a solid, edible object, we know that it can be cut.

The knowledge base plane for domain plans is structured as a hierarchical AND / OR tree. Plans

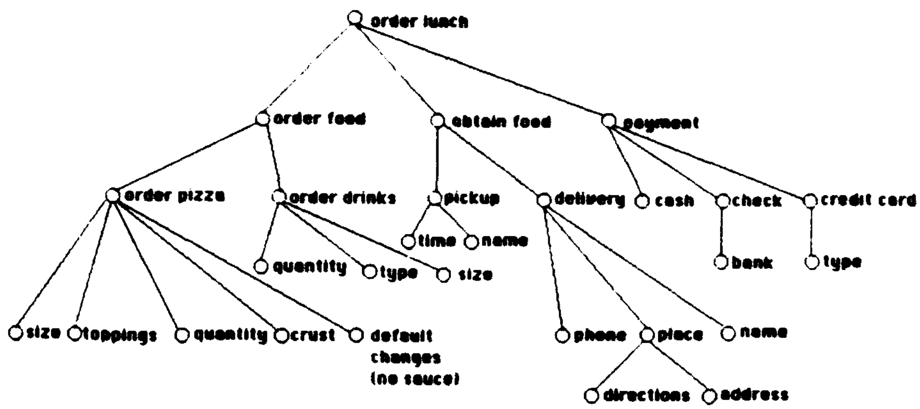

Figure 1: Domain Tree for Pizza Task

are hierarchically organized, allowing for abstractions and least commitment planning. Ordering constraints among plan steps are encoded as both preferences (e.g. order food, then payment and delivery) or required preconditions, and are represented in *control schemas*. We also encode exclusive OR relations among plan steps to represent alternate methods for achieving the same end (e.g. delivery or carry-out). Each node in the plan hierarchy is indexed to a set of actions and objects and attributes involved in satisfying the plan part. These sets of action and object combinations are either "required" or "optional". Further, the plan steps themselves are marked as optional or required for solving the problem. We permit the optionality value of a node to be conditional upon other aspects of the dialog. Finally, general semantic and pragmatic knowledge is associated with each node in the plan tree. This knowledge places the plan step and its associated actions and objects in the context of the overall purpose of the plan step.

The domain tree generated for a domain is **"generic"** and represents all ways of solving any problem in the application domain. Whenever a speaker begins to solve a domain problem, an instance of the tree is generated. The structure of the domain tree varies widely across problems in the same domain. Information presented earlier in a dialog often serves to constrain the rest of the dialog, pruning entire solutions paths from consideration and modifying or constraining the actions and objects associated with yet-to-be-executed plan steps. As the discourse progresses, the tree changes. In processing a dialog, the heuristics keep track of what nodes have been completed, what nodes are active and the relationship of the active nodes to inactive nodes (see Figure 2).

The focus stack is used to keep track of currently active plans and subdialogs. It is a standard push down stack. We also use it to keep track of certain types of subdialogs (e.g. clarifications).

The procedures and algorithms for traversing the domain tree, handling subdialogs and topic switching and keeping track of what is active, what is complete, etc. are domain independent. The domain tree traversal algorithms have been described previously [11, 10]. Here, we overview them to show how they must change to permit discourse plans and subdialogs to be processed. Basically, we deal with both discourse and domain plans by predicting "what can come next" and then match the current utterance representation again the alternate predictions to see which is most closely resembled. We use a single control structure for recognizing domain plans and discourse actions, for constraint propagation and for generating future predictions [12]. The "no frills" tree traversal methodology states:

- do not repeat completed actions
- continue a subtree until it is complete, completing all children nodes, followed by non-excluded sibling nodes, etc.
- propagate constraints as you progress, eliminating subtrees and constraining how a plan step may be realized as constrained by discourse information

The idea is to trace through the tree, hierarchically, pursuing each subtask, in any requisite order until complete. Inapplicable subtrees (due to exclusive OR's) are pruned as they become obsolete. Constraints on how a plan step may be executed are also propagated as they are inferred or entailed by the discourse. The active and yet-to-be-completed tree nodes are used to predict what can come next.

However, the incorporation of discourse plans conditionally modifies the first two "rules". Clarifications may be initiated after each turn but are restricted to either correcting the interpretation of what was just said or to acquiring information about only what was newly presented in the turn. Confirmations, and resultant correction subdialogs can only be initiated when a subtree or subtask is complete, before continuing on to a sibling or parent subtree or node. Their content is restricted to some or all of the content discussed in the preceeding subtask. For example, a confirmation may address any and all aspects of what was ordered in the pizza domain, once the order is complete or once the task is complete. Corrections resulting from confirmations are restricted to the active topic being addressed at that point in time in the correction subdialog. Multiple domain goals and plan failures were handled by the initial algorithms. Further, no modification is made to the constraint propagation algorithms except that external context must be explicitly modelled so that modifications or changes can be anticipated and we predict external environment topic switching.

In sum, the addition of discourse plans only slightly modifies existing domain tree traversal heuristics yet they permit systems to predictively account for subdialog phenomena. The domain tree is traversed as previously [10, 11] with three exceptions. First, the system looks for potential clarifications after each interaction. Second, end of subtree herusitics are modified to also look for confirmations. If an when confirmations are found, the system will anticipate potential correction subdialogs should a confirmation fail. The correction subdialog algorithms substantially follow previously established techniques for processing plan failures. Finally, domain tree traversal is modified to permit environmental topic changes by tracking environmental correlates of plan steps and any user displays. The system hence adds the prediction that whenever the environment changes or will change with the next domain plan part, a topic switch may temporarily interrupt the on-going domain plan.

4. Illustrative Example: Representation and Tracking Subdialogs

To illustrate how processing proceeds, we present the following example of a clarification subdialog in the pizza ordering domain.

1. *Pizza Parlor*
2. I'd like to order a pizza
3. *What size?*
4. What sizes do you have?
5. *Small, medium and large*
6. How big's a medium?
7. *12 slices*
8. A small?
9. *8*
10. OK, I'll take a small

To illustrate how this dialog is processed Figure 2 traces the the current focus stack and the state of the instantiated domain graph as it is modified by processing the utterances in the example dialog. At the start of the dialog, an instance of the complete domain tree (Fig. 1) is generated. The generic tree permits speakers to order different types of food and drinks and has alternate methods for both obtaining and paying for them. The system uses its copy of the tree to mark what is active and what has been completed as we process the dialog. The focus stack keeps track of abandoned topics, current topics and the state of the clarification. It begins empty.

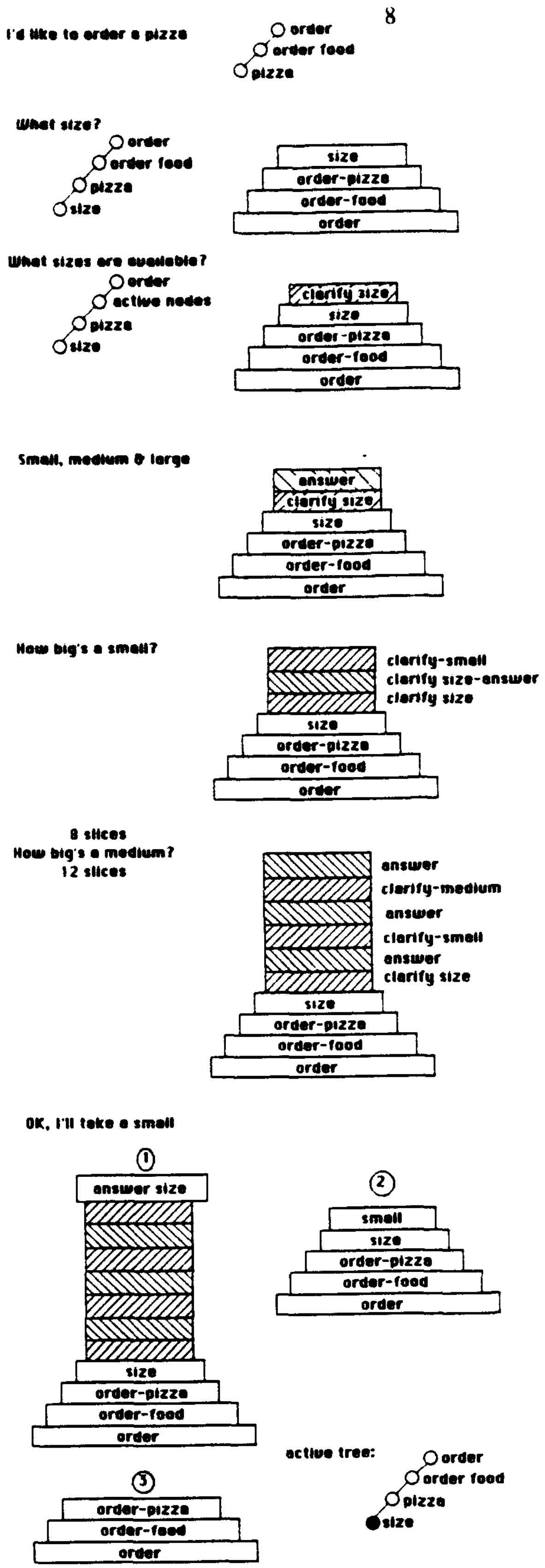

Figure 2: Clarification Dialog Processing Example

After the speaker states *"I'd like to order a pizza"*, the domain tree shows that order, order food and pizza are active nodes. The size question activates the child node of size under pizza. The stack shows order food, order pizza and size, processing the size question as a continuation of current plan. Nothing can be confirmed until a "unit" is complete, for example, the unit of order pizza, including all three required nodes of size, toppings and crust. However, attributes of "pizza-size" are available for clarification. In fact, the next utterance asks just this, to clarify the available values for size. To process this, the system will use only the focus stack. The domain tree will remain unchanged and no information will be propagated to the later dialog from the clarification phase. The answer *"small, medium and large"* introduces three attributes which can be further clarified. Or, the speaker can terminate the subdialog and continue the domain plan by specifying a size. The focus stack shows that the initial clarification has been answered.

Next, the user initiates a second clarification subdialog, requesting further information about the sizes. Potentially, the user could ask individually about each of the attributes (e.g. diameter, slices) of each size (sm, med, lg.) before returning to the main dialog. If the user asked about diameter, it would be to clarify their initial question "How big's a medium" effectively saying "what is the diameter of a medium sized pizza." However, the user is satisfied with the number of slices attribute only asks about two of the sizes and then terminates. At this point, the domain tree records that the size node has been completed and predicts that the dialog will next focus on either the toppings or size nodes. The focus stack records only what is still active, namely order pizza. It should be noticed that the focus stack keeps track of the entire clarification subdialog. It does not pop until the entire subdialog is complete. In this way, we can keep track of what has been completed without modifying the basic domain tree.

References

[1] Allen, J.
Recognizing Intentions from Natural Language Utterances.
In Brady, M. and Berwick, R. C. (editor), *Computational Models of Discourse*, pages 107-164. MIT Press, 1984.

[2] Allen, J. A.
Discourse Structure in the TRAINS Project.
In *Proceedings of the DARPA Speech and Natural Language Workshop*, pages 325-330. February, 1991.

[3] Ferguson, G. and Allen, J. F.
Generic Plan Recognition for Dialogue Systems.
In *Proceedings of the DARPA Human Lanague Technology Conference*. 1993.

[4] Grosz, B.
The Representation and Use of Focus in a System for Understanding Dialogs.
In *IJCAI-79*, pages 67-76. Morgan Kaufmann, 1979.

[5] Grosz, B. J. and Sidner, C. L.
Attention, Intentions and the Structure of Discourse.
Computational Linguistics 12:175-204, 1986.

[6] Litman, D. J. and Allen, J. F.
A Plan Recognition Model for Subdialogs in Conversation.
Cognitive Science 11:163-200, 1987.

[7] Matrouf, K., Gauvin, J.L., Neel, F., Mariani, J.
Adapting Probability-Transitions in DP Matching Process for an Oral Task-Oriented Dialogue.
In *IEEE International Conference on Acoustics, Speech and Signal Processing*. 1990.

[8] Pollack, M.
Plans as Complex Mental Attitudes.
In Cohen, P.R., Morgan, J. and Pollack, M. E. (editors), *Intentions in Communication*. MIT Press, 1990.

[9] Smith, R. W., Hipp, D. R. and Biermann, A. W.
A Dialog Control Alorithm and Its Performance.
In *Proceedings of the Conference on Applied Natural Language Processing*, pages 6. 1992.

[10] Young, S.R., Hauptmann, A.G., Ward, W.H., Smith, E.T., Werner, P.
High Level Knowledge Sources in Usable Speech Recognition Systems.
Communications of the ACM 32(2):183-194, 1989.

[11] Young, S.R.
Use of Dialogue, Pragmatics and Semantics to Enhance Speech Recognition.
Speech Communication 9((5/6)):551-564, 1990.

[12] Young, S. R. and Ward, W. H.
Semantic and Pragmatically Based Re-Recognition of Spontaneous Speech.
In *Proceedings of the European Conference on Speech Communication and Technology*. ESCA: Paris, London, 1993.

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890